CS_411 Critical Systems:

http://www-compsci.swan.ac.uk/~csetzer/lectures/critsys/index.html

Dr. Anton Setzer

http://www-compsci.swan.ac.uk/~csetzer/index.html

Lent Term 2002
A0. Introduction, Overview

(a) An Example of a safety-critical System.

(b) Two aspects of critical systems.

(c) Administrative Issues.

(d) Plan.

(e) Literature
Definition: A critical system is a

- computer, electronic or electromechanical system
- the failure of which may have serious consequences, such as
  - substantial financial losses,
  - substantial environmental damage,
  - injuries or death of human beings.

Definition: A safety-critical system is a

- computer, electronic or electromechanical system
- the failure of which may cause injury or death to human beings.
Some Areas with Critical Systems

- Safety-Critical
  - Medical Devices.
  - Aerospace
    - Civil aviation.
    - Military aviation.
    - Space travel.
  - Chemical Industry.
  - Nuclear Power Stations.
  - Traffic control.
    - Railways.
    - Air traffic.
    - Road traffic (esp. traffic lights).
  - Other military equipment.

- Critical but not safety-critical.
  - Financial institutions.
  - Trading (internet!!).
  - Areas where secrecy is required.
    - Defense.
    - Secret service.
    - Sensitive areas in companies.
Åsta Train Accident (January 5, 2000)

Report from November 6, 2000
http://odin.dep.no/jd/norsk/publ
/rapporter/aasta/
Figur 3.1: Rørosbanens beliggenhet i Norge. Rørosbanen markert med rødt
Kilde: Jernbaneverket
Figur 3.5: Rørosbanen Hamar-Støren med stasjoner, regiongrenser og grense mellom de områder togledersentralene overvåker
Kartgrunnlag: Jernbaneverket
Figure 2-3. Rudstad station track layout
Rena
Train 2302
75 passengers

Rudstad
Train 2369
10 passengers

**Sequence of Events:**

- Railway with one track only. Therefore crossing of trains only at stations possible.

- According to timetable crossing of trains at Rudstad.

- Train 2302 is 21 minutes behind schedule. When reaching Rena, delay is reduced to 8 minutes. Leaves Rena after a stop with green exit signal 13:06:15, in order to cross 2369 at Rudstad.

- Train 2369 leaves after a brief stop Rena 13:06:17, 3 minutes ahead of timetable, probably in order to cross 2302 at Rena.
Rena
Train 2302
75 passengers

Rudstad
Train 2369
10 passengers

- Local train shouldn’t have had green signal.

- **13:07:22** Alarm signaled to the rail traffic controller (no audible signal).

- Rail traffic controller sees alarm approx. **13:12**.

- Traffic controller couldn’t warn trains, because of use of mobile telephones (the correct number hadn’t been passed on to him).

- Trains collide 13:12:35, 19 persons are killed.
Figur 3.12: De to togsettene etter at brannen var slokket
Foto: Politiet/Kripos

Lokomotivet på sydgående tog som veltet ved kollisjonen

Vogn 3 i sydgående tog som ble boyd i j-form

Figur 3.13: Motorvogn, lokomotiv, vogn nr.3 og nr. 2 etter at brannen var slokket
Foto: Politiet/Kripos
Investigations

- No technical faults of the signals found.
- Train driver was not blinded by sun.
- 4 incidents of wrong signals with similar signaling systems reported:
  - Exit signal green and turns suddenly red. Rail controller says, he didn’t give an exit permission.
  - Hanging green signal.
  - Distant signal green, main signal red, train drives over main signal, and pulls back. Rail controller surprised about the green signal.
  - **18 April 2000** Train has green exit signal. When looking again, notices that the signal has turned red. Train controller hasn’t given exit permission.
- Several safety-critical deficiencies in the software found (some known before!)
- The software used was completely replaced.
The status printouts tell nothing about short breaks or transients on the mains. There are no reports from other sensitive power consumers about such problems either.

3.5.13 Deficiencies

Deficiencies in the NSB-87 system are listed in Appendix D. The most important deficiencies are repeated below. SINTEF has tagged some of the deficiencies as safety critical and performed a preliminary ranking according to expected criticality.

**Known deficiencies** (Ref. Appendix D.1)

1. In this system commands for intersection route locking and route locking time delay are stored as parameters in the CTC. If the interface PLC in the CTC fails, train route commands can be faulty transmitted to the safety systems without time delay. In principle, the safety systems cannot prevent the setting of hostile train routes. An exit train route can be secured before the arriving train has come to a full stop. In this event the total safety is based on the safety of the CTC interface PLC, not on the safety of the local signalling system. **This is a safety critical event.**

2. Missing red light control will release Tsp M/O (Drawing 21, Ref. [1]). A line block request signal from the neighbouring station will cause BkM+ to pull up. In this case only two separate PLC output contacts prevent the green exit signal from lighting. In worst case conditions this deficiency could give a wrong signal aspect of up to 3 sec duration (see Sections 5.3.1 and 7.2.1). A repeater indicator function *(gjentakellessepper*) implemented with a safety relay in the BkM+ control circuit similar to that used in NSI-63 should prevent the safety being dependent on the PLC contacts only. **This is a safety critical event.**

3. In the NSB-87 safety system concept it is possible to set entry to one track and exit from another in the same direction when dependency on a road security system exists and the road barrier has not been moved into the lowered position. (Drawing 20, Ref. [1]). **This is a safety critical event.**

4. For exit train routes the PLC is to prevent the setting of hostile train routes in the same direction (Drawing 21, Ref. [1]). A specific function, repeater indicator *(gjentakellessepper*), is implemented in the PLC program to secure that only one train can be on the line block at a time. **This is a safety critical function.**

5. In this type of safety systems a short track section should be installed outside the entry signals integrated in the logic for line block control. The main reason for this recommendation is to prevent the setting of conflicting green exit signals on neighbouring stations during shunting operations. **This is a safety critical deficiency.**

**Deficiencies found during the investigations** (Ref. Appendix D.2)

1. Refer to Sec. D.1, Paragraph 2 in the list. A similar deficiency was also identified for the opposite driving direction, i.e. from south to north. Missing the red light control will release Tsp L/N (Drawing 23, Ref. [1]). A line block request signal from the neighbouring station (Rena) will cause BkL+ to pull up. In worst case conditions this deficiency could give a wrong signal aspect of up to 3 sec duration. **This is a safety critical event.**

2. In the NSB-87 simplified line block design the line block control relays Bk(M/L)+ serves two different system functions. First, serving the line block request. Second, setting and locking the line block function. This specific system design definitely reduces the safety and reliability levels of the line block system. In the NSI-63 system concept these functions are served by different relays. **This should be characterised as a safety critical system deficiency.**
• **SINTEF** (Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology) found no mistake leading directly to the accident. **Conclusion of SINTEF:** No indication of abnormal signal status. 
  ⇒ Mistake of train driver (died in the accident). (Human Error).

• Assessment of report by **Railcert Criticism** of SINTEF’s conclusion:
  - “We feel that Sintef’s conclusions 2,4 and 6 suggest that a technical cause, related to the signaling installations, for the accident can be ruled out. We support this conclusion only inasmuch as it applies to a **steady state, single cause failure**. We do however stress the need to look beyond such “simple causes” ”.
• “A number of known reports of anomalies in similar installations exist. Based on these, we have been able to construct theoretical scenarios where the behaviour of the signaling installations might at least have contributed to the cause of the accident.”
Conclusion

- It is possible that the train driver of the local train was driving against a red signal.

- The fact that he was stopping and left almost at the same time as the other train and 3 minutes ahead of time, makes it likely that he received an erroneous green exit signal due to some software error.

It could be that the software under certain circumstances when giving an entrance signal into a block, for a short moment gives the entrance signal for the other side of the block.

One possible reason for that could be a level road crossing in between.

- Even if this particular accident was not due to a software error, apparently this software has several safety-critical errors.
• In the protocol of an extremely simple installation (Brunna, 10 km from Uppsala), which was established 1957 and exists in this form in 40 installations in Sweden, a safety-critical error was found when verifying it with a theorem prover formally 1997.

• Lots of other errors in the Swedish railway system were found during formal verification process.
Lessons to be Learned

Many of the following are typical for such accidents:

- Cost cutting precedes accidents.

- Warning signs (safety critical events) are ignored. Terminology: **incidents** or **near-misses** = almost accidents.

- Sooner or a later an accident was about to happen. Principle of **root cause**:
  - Root cause of an accident = weakness, that not only contributed to that particular accident but as well affect future accidents:
    * Cultural attitudes (low level of awareness of security issues).
    * Organizational attitudes.
    * General technical problems (that the software used is too complex in order to guarantee correctness).
  - Root causes result, if no counter measures are taken, sooner or later to accidents.
  - It is important to identify root causes of accidents and incidents and carry out measurements against them in order to prevent future accidents.
Lessons to be Learned (Cont.)

- Accidents have usually multiple causes.

- A sequence of events had to happen in order for the accident to take place.
  
  - Preliminary events.
    
    = events which influence the initiating event. without them the accident cannot advance to the next step (initiating event).
    In the main example:
    * Express train is late. Therefore crossing of trains first moved from Rudstad to Rena.
    * Delay of the express train reduced. Therefore crossing of trains moved back to Rudstad.
  
  - Initiating event, trigger event.
    Mechanism that causes the accident to occur.
    In the main example:
    * Both trains leave their stations on crash cause, maybe caused by both trains having green signals.
- **Intermediate events.**
  Events that may propagate or ameliorate the accident.
  * **Ameliorating events** can prevent the accident or reduce its impact.
  * **Propagating events** have the opposite effect.
  In the example:
  * No system installed which stops a train passing over a red signal (if it was red).
  * Rail traffic controller is busy with traffic on another railway line.
  * Warning light appears.
  * Rail traffic controller doesn’t see warning light.
  * Rail traffic controller doesn’t have mobile number.
  * Diesel engines started to burn.

- When designing safety critical systems, **one should**
  - avoid triggering events
  - add additional safeguards, which prevent a triggering event from causing an accident or reduces its impact.
Lessons to be Learned (Cont.)

- Accidents often reveal general negligence of safety procedures:
  - Not properly designed software.
  - Use of standard mobile telephones without any special arrangements.

- Investigations
  - don’t search for the real reason (root cause), but for somebody to blame (the driver).

- Usually at the end of an investigation conclusion “human error”.

- The architecture of the software was investigated but no detailed search for a bug was done. (Note that it is very difficult to get protocols for even simple situations correct.)

- Afterwards, concentration on the trigger events, but not much attention to preliminary and intermediate events.
(b) Two Aspects of Critical Systems

- **Software engineering aspect.**
  - General safety procedures.
  - Good practices.
  - Development and installation of software with strict control of safety and quality control.
  - Documentation
- System aspect.
  - Computer system.
  - Hardware connected (hardware failure in sensors, relays and computers)
  - Interaction with environment.
  - Human-machine interaction.
- Risk and hazard-analysis.
- Risk management.
- Validation and testing.
- A lot to be learned from other industries.
  (Esp. chemical industry, nuclear power industry, aviation).
Two aspects (Cont.)

- **Tools for generating correct software.**
  - Software bugs can not be avoided by careful design.
    - Especially with distributed algorithms.
  - Need for verification techniques using formal methods.
  
  Different levels of rigour:
  - Application of formal methods by hand, without machine assistance.
  - Use of formalized specification languages with some mechanized support tools.
  - Use of fully formal specification languages with machine assisted or fully automated theorem proving.
Two Streams in this Module

• Stream A (one session per week):
  – Software engineering aspect + general overview over formal methods.
  – Industrial practices.

• Stream B (one session per week):
  – Closer look at one prototype example of a tool which allows to write 100% correct software.
  – Programming with dependent types, based on Martin-Löf type theory.
  – Still area of research.
  – Use of the theorem prover Agda.
    * This part will be heavily machine based.
    * Experimental system.
    * Application of its ideas not limited to safety critical software.
      · Dependent types will sooner or later be used in ordinary programming languages.
        (Templates in C++ or soon in Java (??) is one approximation.)
  – Goal is that students have been in contact with one proof assistant.
(c) Administrative Issues

Address:
Dr. Anton Setzer
Dept. of Computer Science
University of Wales Swansea
Singleton Park
SA2 8PP
UK

Room    Room 211, Faraday Building
Tel.     (01792) 513368
Fax.     (01792) 295651
Email    a.g.setzer@swansea.ac.uk
Assessment and Timetable

• Assessment
  – 80% written examination in May.
  – 20% coursework:
    – 4 small assignments. Each counts 5%.
      * Handed out approx. every 2nd week.
      * Due two weeks later.

• Two lectures per week.
  – Thursday, 10:00, Robert Recorde Room
  – Friday, 12:00, Robert Recorde Room
Course Material

- Web page contains overhead slides from the lectures. Course material will be continually updated.

- Course material always available from student office in the department of computer science, room 206.

- Please don’t print out lecture notes on departmental printers – ask secretaries for copies. Help to save the tax payers money!! – Cost for printing is 10 times the cost of photo copying.
(d) Plan

- Learning outcome:
  - Familiarity with issues surrounding safety-critical systems.
  - Understanding of techniques for specifying and verifying high-integrity software.
  - Experience with one proof-assistant.

- Plan Stream A:
  
  A0. Introduction, overview.
  A2. Hazard analysis.
  A3. Risk analysis.
  A4. Developing safety-critical systems.
  A5. System reliability.
  A6. Design methods and tools.
  A7. Formal methods.
  A8. Verification, validation, testing.
Plan (Cont.)

- Plan Stream B:
  B1. Introduction.
  B2. Propositions as types.
  B3. The logical framework.
  B4. Data types.
  B5. Interactive programs in dependent type theory.
  B6. Case studies.
(e) Literature

Books Relevant for Stream A

- Main course book:

- Supplementary books on software-engineering aspects:
• Some books on general formal methods:
  
    Reference to medical software using a widely used specification language. Probably not to be treated here.
  
    B-method is a specification language, which might replace Z in the future.
  
  - John Barnes: *High integrity Ada. The SPARK approach*.
    Use of a subset of Ada with proof annotations in order to develop secure software. Developed and used in industry.
  
    Introduction to logic. Covers model checking, an industrial method, very well. But probably not to be treated here.


• Aarne Ranta: *Type-theoretic grammar*. Clarendon Press, 1995. Use of type theory in linguistics and for translation between languages. Supposed to have a good and simple introduction into type theory.